

## Rezolvări atestat 2012

1. Fișierul *date.in* conține pe prima linie o valoare naturală nenulă  $n$ , iar pe următoarele  $n$  linii, separate printr-un spațiu, câte o pereche de numere naturale nenule reprezentând numărătorul, respectiv, numitorul unei fracții ( $2 \leq n \leq 10$ ). Să se determine suma acestor fracții. Numărătorul și numitorul fracției obținute în forma ireductibilă (valori mai mici sau egale decât 1000000000) vor fi scrise în fișierul *date.out* pe prima linie a acestuia separate printr-un spațiu.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	4	43 20	$\frac{7}{6} + \frac{1}{3} + \frac{1}{4} + \frac{2}{5} = \frac{43}{20}$
	7 6		
	1 3		
	1 4		
	2 5		

```
#include<fstream.h>
#include<iostream.h>
int cmmdc(int d, int i)
{
    if(i==0)
        return d;
    else
        return cmmdc(i,d%i);
}
int cmmmc(int a,int b)
{
    return (a*b)/cmmdc(a,b);
}
int main( )
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int a,b,c,d,n;
  fin>>n;
  fin>>a;
  fin>>b;
  for (int i=2;i<=n;i++)
  {
      fin>>c;
      fin>>d;
      a=a*cmmmc(b,d)/b + c*cmmmc(b,d)/d;
      b=cmmmc(b,d);
  }

  for (d=2;d<=a/2;d++)
      {
          if (a%d==0)
              if (b%d==0)
                  {
                      a=a/d;
                      b=b/d;
                  }
          sau fout<<a/cmmdc(a,b); fout<<b/cmmdc(a,b);
      }
```

```

    }
    }
    fout<<a<<" "<<b<<endl;
    fin.close();
    fout.close();
    system("pause");
}

```

2. Fișierul *date.in* conține pe unica sa linie un număr natural  $n$  ( $n \leq 1000000000$ ). Să se determine „cifra de control” a acestui număr, efectuând suma cifrelor sale, apoi suma cifrelor acestei sume, etc., până se obține o sumă formată dintr-o singură cifră. Rezultatul obținut va fi scris în fișierul *date.out*.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	1971	9	$1971 \rightarrow 18 \rightarrow 9$

```

#include <iostream.h>
#include<fstream.h>
int sc(int x)
{
    if(x<10)
        return x;
    else
        return sc(x/10)+x%10;
}
int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int x,s;
  fin>>x;
  while(x>10)
      x=sc(x);
  fout<<x<<endl;
  fin.close();
  fout.close();

  system("pause");
}

```

3. Fișierul *date.in* conține pe unica sa linie, separate prin spații, trei numere naturale nenule,  $a$ ,  $b$  și  $n$ ,  $0 \leq a < b \leq 10000$ ,  $0 < n \leq 10000$ . Să se determine primele  $n$  zecimale ale fracției  $a/b$ . Cifrele determinate vor fi scrise în fișierul *date.out* pe o singură linie.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	5 23 5	21739	$\frac{5}{23} = 0.21739\dots$

```

#include <iostream.h>
#include <iomanip.h>
#include<fstream.h>
int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);

```

```

        int a,b,n;
    fin>>a;
    fin>>b;
    fin>>n;
    fout<<setprecision(n)<<((float)a/b*pow(10,n);
    fin.close();
    fout.close();
    system("pause");
}

```

4. Să se determine toate numerele naturale prime de trei cifre care citite invers sunt tot numere prime. Rezultatele obținute vor fi scrise în fișierul *date.out*, câte unul pe fiecare linie a fișierului.

Ex.	<i>date.out</i>	Explicație
	101	două astfel de numere sunt 101 și 167
	...	
	167	
	...	

```

#include<fstream.h>
# include <iostream.h>
# include <math.h>
int ogl(int x)
{
    int o=0, c;
    while(x!=0)
    {
        c= x%10;
        o=o*10+c;
        x=x/10;
    }
    return o;
}
int prim(int x,int d=2)
{
    if(d>sqrt(x))
        return 1;
    else
        if(x%d==0)
            return 0;
        else
            return prim(x,d+1);
}
int main()
{fstream fout("date.out",ios::out);
    int x;
    for(x=100;x<1000;x++)

```

```

    {
        if(prim(x)==1 && prim(ogl(x))==1)
            fout<<x<<" ";
    }
    fout.close();
    system("pause");
}

```

5. Un număr natural  $n$  este deosebit dacă există un număr natural  $m$  cu proprietatea că adunând acest număr  $m$  cu suma cifrelor numărului  $m$  se obține numărul  $n$ . Fișierul *date.in* conține pe unica sa linie un număr natural  $n$  ( $n \leq 1000000000$ ). Să se verifice dacă acest număr este deosebit. Rezultatul verificării va fi scris în fișierul *date.out* pe unica linie a acestui fișier sub forma unui mesaj corespunzător (DA sau NU după cum numărul are proprietatea respectivă sau nu).

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	1235	DA	1235 poate fi scris ca 1225+10
	<i>date.in</i>	<i>date.out</i>	
	20	NU	

```

#include<fstream.h>
# include <iostream.h>

int sc(int x)
{
    if(x<10)
        return x;
    else
        return sc(x/10)+x%10;
}

int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int n;
  fin>>n;
  if(n-sc(n)>0)
      fout<<"DA";
  else
      fout<<"NU";
  fin.close();
  fout.close();
  system("pause");
}

```

6. Fișierul *date.in* conține pe unica sa linie un număr natural nenul  $n$  reprezentând numărul de pagini ale unei cărți ( $20 \leq n \leq 10000$ ). Să se determine câte cifre au fost folosite la paginarea cărții. Rezultatul obținut va fi scris în fișierul *date.out*.

Ex.	<i>date.in</i>	<i>date.out</i>
	100	192

<i>date.in</i>	<i>date.out</i>
15	21

```
#include<fstream.h>
# include <iostream.h>
int v[10];
int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int i,s=0,n,aux,c;
  fin>>n;
  for(i=1;i<=n;i++)
  {
    aux=i;
    while(aux!=0)
    {
      c=aux%10;
      v[c]=v[c]+1;
      aux=aux/10;
    }
  }
  for(i=0;i<=9;i++)
    s=s+v[i];
  fout<<s;

  fin.close();
  fout.close();
  system("pause");
}
```

7. Fișierul *date.in* conține pe unica sa linie, separate printr-un spațiu, două numere naturale nenule,  $a$  și  $b$  ( $1 \leq a, b \leq 1000000000$ ). Să se verifice dacă aceste numere pot fi termeni consecutivi ai șirului Fibonacci (1,1,2,3,5,8,...). Rezultatul verificării va fi scris în fișierul *date.out* pe unica linie a acestui fișier sub forma unui mesaj corespunzător (DA sau NU după cum cele două numere sunt termeni consecutivi ai șirului Fibonacci sau nu).

Ex.	<i>date.in</i>	<i>date.out</i>
	2 3	DA

<i>date.in</i>	<i>date.out</i>
89 55	DA

<i>date.in</i>	<i>date.out</i>
4 5	NU

```
.
#include<fstream.h>
# include <iostream.h>
int fibo(int a, int b)
```

```

{
    if(a==1 && b==1)
        return 1;
    else
        if(a>b)
            return 0;
        else
            return fibo(b-a,a);
}
int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int a,b;
  fin>>a>>b;
  if(fibo(a,b))
      fout<<"DA";
  else
      fout<<"NU";
      fin.close();
  fout.close();
  system("pause");
}

```

8. Fișierul *date.in* conține pe unica sa linie un număr natural  $n$  ( $n \leq 1000000000$ ). Să se determine toate reprezentările posibile ale acestui număr ca sumă de numere naturale nenule consecutive. Aceste reprezentări vor fi scrise în fișierul *date.out*, câte una pe fiecare linie a fișierului; numerele din cadrul fiecărei reprezentări vor fi separate prin spații.

Ex.	<i>date.in</i>	<i>date.out</i>
	15	1 2 3 4 5 4 5 6 7 8

<i>date.in</i>	<i>date.out</i>
50	8 9 10 11 12 11 12 13 14

```

#include<fstream.h>
# include <iostream.h>

int main()
{ fstream fin("date.in",ios::in), fout("date.out",ios::out);
  int j,n,s,k;
  fin>>n;
  for(int i=1;i<=n/2;i++)
  {
      s=i;
      k=i+1;
      while(s<n)
          s=s+k++;
  }
}

```

```

        if(s==n)
        {
            for(j=i;j<k;j++)
                fout<<j<<" ";
            fout<<endl;
        }
    }

    fin.close();
    fout.close();
    system("pause");
}

```

9. Un număr natural se numește „super prim” dacă numărul respectiv și toate prefixele acestuia sunt numere prime. Fișierul *date.in* conține pe unica sa linie un număr natural  $n$  ( $n \leq 1000000000$ ). Să se verifice dacă acest număr este „super prim” și, în caz afirmativ, să se determine prefixele sale. Rezultatul verificării va fi scris în fișierul *date.out* pe prima linie a acestui fișier sub forma unui mesaj corespunzător (DA sau NU după cum numărul este „super prim” sau nu); dacă numărul este „super prim”, fiecare următoare linie a fișierului va conține câte un prefix al numărului respectiv.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	239	DA	numărul 239 este prim; prefixele sale (2 și 23) sunt numere prime
		2	
		23	
	<i>date.in</i>	<i>date.out</i>	Explicație
	17	NU	numărul 17 este prim; prefixul său (1) nu este prim

```

#include<fstream.h>
# include <iostream.h>
# include <math.h>
int prim(int x,int d=2)
{
    if(d>sqrt(x))
        return 1;
    else
        if(x%d==0)
            return 0;
        else
            return prim(x,d+1);
}
int main()
{fstream fin("date.in",ios::in), fout("date1.out",ios::out);
    int n,ok=1,aux;
    fin>>n;
    aux=n;
    while(n!=0)
    {

```

```

        if(prim(n)==0)
            ok=0;
        n=n/10;
    }
    if(ok==1)
    {
        fout<<"da";
        while(aux!=0)
        { if (aux!=n)
        { fout<<aux<<" ";
            aux=aux/10;
        }
        }
    }
    else
        fout<<"nu";
    fin.close();
    fout.close();
    system("pause");
}

```

10. Fișierul *date.in* conține pe unica sa linie un număr natural  $n$  ( $n \leq 10000000000$ ). Să se determine cel mai apropiat număr prim față de  $n$ . Programul va conține cel puțin un subprogram, iar rezultatul obținut va fi scris în fișierul *date.out*. Dacă sunt două numere prime egal departate de  $n$ , se poate afișa oricare dintre ele.

Ex.	<i>date.in</i> 1400	<i>date.out</i> 1399	Explicație 1399 este numărul prim cel mai apropiat de 1400
	<i>date.in</i> 3019	<i>date.out</i> 3019	Explicație 3019 este număr prim
	<i>date.in</i> 3000	<i>date.out</i> 3001	Explicație 3001 și 2099 sunt numere prime egal departate de numărul 3000

```

#include<iostream.h>
#include<math.h>
#include<fstream.h>

```

```

int prim(int x,int d=2)
{
    if(d>sqrt(x))
        return 1;
    else
        if(x%d==0)
            return 0;
        else
            return prim(x,d+1);
}

```

```

int nr(int n)
{
    int a=n, b=n;
    while(!prim(a))
        a--;
    while(!prim(b))
        b++;
    if(n-a<=b-n)
        return a;
    else
        return b;
}
fstream fin("date.in",ios::in), fout("date1.out",ios::out);
int main()
{
    int n,x;
    fin>>n; x=nr(n);
    fout<<x;
    fin.close();
    fout.close();
    system("pause");
}

```

11. Fișierul *date.in* conține pe prima sa linie un număr natural  $n$  ( $1 \leq n \leq 100$ ), iar pe următoarea linie, separate prin spații, cele  $n$  elemente ale unui vector de numere naturale. Să se determine câte din elementele vectorului dat sunt numere Fibonacci. Rezultatul obținut va fi scris în fișierul *date.out*.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	10	5	sunt cinci numere Fibonacci în vectorul dat;
	5 10 1 7 9 8 1 6 55 19		acestea sunt: 5, 1, 8, 1, 55

```

#include <iostream.h>
#include <fstream.h>
int fibo(int x)
{
    int tv=1,tn=1,ok=0;
    while(tn<x)
    {
        tn=tv+tn;
        tv=tn-tv;
    }
    if(tn==x)
        return 1;
    else
        return 0;
}
int main()
{ fstream fin("date.in",ios::in),fout("date.out",ios::out);
  int x[30],n,cate=0;
  fin>>n;

```

```

for(int i=1;i<=n;i++)
{
    fin>>x[i];
    if(fibo(x[i]))
        cate++;
}
fout<<cate;
fin.close();
fout.close();
system("pause");
}

```

12. Fișierul *date.in* conține pe prima sa linie un număr natural  $n$  ( $1 \leq n \leq 100$ ), iar pe următoarea linie, separate prin spații, cele  $n$  elemente ale unui vector de numere naturale. Să se verifice dacă vectorul dat este o mulțime (în sensul cunoscut din matematică), dacă nu, să se transforme acest vector în mod corespunzător. Elementele vectorului rezultat se vor afișa pe prima linie a fișierului *date.out*, separate prin spații.

Ex.	<i>date.in</i>	<i>date.out</i>	Explicație
	10	1 2 9 4 6 5 20 3	elementele vectorului inițial nu sunt două câte două distincte
	1 2 9 4 2 6 5 1 20 3		
	<i>date.in</i>	<i>date.out</i>	Explicație
	7	1 2 9 4 6 5 3	elementele vectorului inițial sunt două câte două distincte
	1 2 9 4 6 5 3		

```

#include <iostream.h>
#include <fstream.h>
void sterge(int x[],int &n,int k)
{
    int i;
    for(i=k;i<n;i++)
        x[i]=x[i+1];
    n--;
}
int apare(int x[],int y,int k,int n)
{
    for(int i=k+1;i<=n;i++)
        if(x[i]==y)
            return 1;
    return 0;
}
int main()
{
    fstream fin("date.in",ios::in),fout("date.out",ios::out);
    int x[30],n,i,ok=1;
    fin>>n;
    for(i=1;i<=n;i++)
    {
        cout<<"x["<<i<<"]="<<endl;
        fin>>x[i];
    }
    for(i=1;i<=n;i++)
    {

```

```

        if(apare(x,x[i],i,n))
        {
            sterge(x,n,i);
            i--;
        }
    }
    for(i=1;i<=n;i++)
        fout<<x[i]<<" ";
    fin.close();
    fout.close();
    system("pause");
}

```

13. Fișierul *date.in* conține: pe prima sa linie un număr natural  $n$  ( $1 \leq n \leq 100$ ), pe a doua linie, separate prin spații, cele  $n$  elemente ale unui vector  $a$  de numere întregi, pe a treia linie un număr natural  $m$  ( $1 \leq m \leq 100$ ), iar pe a patra linie, separate prin spații, cele  $m$  elemente ale unui vector  $b$  de numere întregi. Să se afișeze pe prima linie a fișierului *date.out* câte din elementele vectorului  $b$  sunt strict mai mici decât toate elementele vectorului  $a$ .

<p>Ex. <i>date.in</i></p> <pre> 10 -3 2 9 4 2 6 -5 -1 20 3 8 6 -5 0 -6 20 2 -18 -6 </pre>	<p><i>date.out</i></p> <pre> 3 </pre>	<p>Explicație</p> <p>În al doilea vector sunt trei elemente strict mai mici decât toate elementele primului vector; acestea sunt: -6, -18, -6</p>
---	---------------------------------------	---

```

#include <iostream.h>
#include <fstream.h>
int min(int x[],int n)
{
    int m=x[1];
    for(int i=2;i<=n;i++)
        if(m>x[i])
            m=x[i];
    return m;
}
int main()
{
    fstream fin("date.in",ios::in),fout("date.out",ios::out);
    int x[30],y[30],m,n,mn,cate=0,i,j;
    fin>>n;
    for(int i=1;i<=n;i++)
        fin>>x[i];
    fin>>m;
    for(j=1;j<=m;j++)
    {
        fin>>y[j];
    }

    for(j=1;j<=m;j++)
        if(y[j]<min(x,n))
            cate++;
    fout<<cate;
    fin.close();
}

```

```

        fout.close();
system("pause");

}

```

14. În fișierul *date.in* se găsesc două numere mari (care pot avea mai mult de 10 cifre), câte unul pe o linie. Să se afișeze pe prima linie a fișierului *date.out* suma celor două numere.

Ex.	<i>date.in</i>	<i>date.out</i>
	45899200768797	45899251768664
	50999867	

```

#include <fstream.h>
#include <iomanip.h>
#include <conio.h>
int x[20];
int main()
{
    int t,s,lx=0;
    long a,b,r;
    fstream f("date.txt",ios::in);
    f>>a;
    f>>b;
    while(a!=0 && b!=0)
    {
        r=a%10+b%10+t;
        lx++
        x[lx]=r%10;
        t=(a%10+b%10)/10;
        a=a/10;
        b=b/10;
    }
    while(a>0)
    {
        r=a%10+t;
        t=r%10;
        lx++
        x[lx]=r%10;
        a=a/10;
    }
    while(b>0)
    {
        r=b%10+t;
        t=r%10;
        lx++
        x[lx]=r%10;
        b=b/10;
    }
    for(int i=lx;i>=1;i--)
        cout<<x[i];
    getch();
}

```

```

        return l;
    }

```

15. Se considera un vector cu n componente. Stiind ca el contine doua subsecvente de numere ordonate crescator , sa se ordoneze intregul vector prin interclasarea celor doua subsecvente.

```

#include <iostream.h>
#include "functii.txt"
void interclasare(int r[],int &l,int x[],int lx,int y[],int ly)
{
    int i=1,j=1;
    l=0;
    while(i<=lx && j<=ly)
    {
        if(x[i]<=y[j])
        { l++;
          r[l]=x[i++];}
        else { l++;
              r[l]=y[j++];}
    }
    while(i<=lx)
    { l++;
      r[l]=x[i++];}
    while(j<=ly)
    { l++;
      r[l]=y[j++];}
}
void main()
{
    int x[30],n,l,i,r[30],y[30],ly=0,z[30],lz=0;
    citeste(x,n);
    for(i=1;i<n;i++)
        if(x[i]>x[i+1])
        { ly++;
          y[ly]=x[i];}
        for(int j=i+1;j<=n;j++)
        { lz++;
          z[lz]=x[j];}
    }
    else
    { ly++;
      y[ly]=x[i];}
    }
    scrie(y,ly);
    cout<<endl;
    scrie(z,lz); cout<<endl;
    interclasare(r,l,y,ly,z,lz);
    scrie(r,l);
    cout<<endl;
}

```

16. Se citeste de la tastatura un tablou unidimensional cu n elemente numere intregi. Sa se afiseze elementul care apare de cele mai multe ori in tablou. Daca exista mai multe astfel de elemente, se vor afisa toate.

Ex. Pentru n=8 si elementele (23,7,11,7,19,7,11,11) se vor afisa elementele 7 si 11, care apar fiecare de cate 3 ori.

```
# include <iostream.h>
# include "functii.txt"
int apare(int x[],int n,int k)
{
    int ap=0;
    for(int i=1;i<=n;i++)
        if(x[i]==k)
            ap++;
    return ap;
}
void sterge(int x[],int n,int k)
{
    for(int i=1;i<=n;i++)
    {
        if(x[i]==k)
        {
            for(int j=i;j<n;j++)
                x[j]=x[j+1];
            n--;
        }
    }
}
int max(int x[],int n)
{
    int mx=x[1];
    for(int i=2;i<=n;i++)
        if(x[i]>mx)
            mx=x[i];
    return mx;
}
void main()
{
    int x[50],j,n,ap[50],max,i;
    citeste(x,n);
    for(i=1;i<=n;i++)
        ap[i]=apare(x,n,x[i]);
    for(i=1;i<=n;i++)
        if(max(ap,n)==ap[i])
        { cout<<x[i]<<" ";
          for(j=i;j<=n;j++)
              { if(x[i]==x[j])
                  { sterge(x,n,x[j]);

```

```

        sterge(x,n,x[i]);
    }
}
}

```

17. Fie  $v$  un vector de numere intregi. Sa se construiasca un vector  $w$ , astfel incat  $w[i]$ =numarul de aparitii ale lui  $v[i]$  in vectorul  $v$ . Sa se afiseze cei doi vectori, fiecare pe o linie.

Ex: pentru  $v=(1,5,2,1,5,7,2,1,5)$  se obtine  $w=(3,3,2,3,3,1,2,3,3)$

```

#include <iostream.h>
#include "functii.txt"
int apare(int x[],int n,int k)
{
    int ap=0;
    for(int i=1;i<=n;i++)
        if(x[i]==k)
            ap++;
    return ap;
}
void main()
{
    int x[30],n,ap[30],i;
    citeste(x,n);
    for(i=1;i<=n;i++)
        ap[i]=apare(x,n,x[i]);
    cout<<"sirul dat: ";
    scrie(x,n);
    cout<<endl;
    cout<<"Sirul aparitiilor: ";
    scrie(ap,n);
    cout<<endl;
}

```

18. Sa se construiasca o matrice  $A$  cu  $n$  linii si  $n$  coloane ce se completeaza cu termenii lui Fibonacci. Completarea se va face pe linii . Nu se vor folosi structuri de date auxiliare.

Ex. Pentru  $n=3$  se va afisa matricea

```

1  1  2
3  5  8
13 21 34.

```

```

#include <iostream.h>
#include "functii.txt"
int fibo(int k)
{
    int i, tn,tv;
    tn=tv=1;
    i=1;
    if(k==1 || k==2)

```

```

        return 1;
    for(i=3;i<=k;i++)
    {
        tn=tv+tn;
        tv=tn-tv;
    }
    return tn;
}
void main()
{
    int k, n, i, j, a[20][50];
    cout<<"n="; cin>>n;
    k=1;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            a[i][j]=fibo(k);
            k++;
        }
    scrieM(a,n);
}

```

19. Se da o matrice de ordinul n. Se considera ca diagonalele sale impart matricea in 4 zone: nord, sud, vest si est. Se cere sa se calculeze suma elementelor impare din zona de nord a matricei.

```

#include <iostream.h>
#include "functii.txt"
void main()
{
    int a[50][50], n, i, j, s=0, st, dr;
    citesteMP(a,n);
    st=2;
    dr=n-1;
    for(i=1;i<=n/2-(n%2==0);i++)
    {
        for(j=st;j<=dr;j++)
            if(a[i][j]%2!=0)
                s=s+a[i][j];
        st++;
        dr--;
    }
    scrieMP(a,n);
    cout<<s;
}

```

20. Se citește de la tastatură un cuvânt de lungime cel mult 20 de caractere, format numai din litere mari. Să se afișeze toate cuvintele distincte ce se pot forma prin eliminarea câte unui singur caracter din cuvântul dat.

Ex. Pentru cuvântul BINE se vor afișa , nu neapărat în această ordine , cuvintele: INE, BNE, BIE, BIN .

```

#include <iostream.h>
#include <string.h>
void main()
{
    char x[21],aux[21];
    int i;
    cout<<"cuvantul: "; cin>>x;
    strcpy(aux,x);
    for(i=0;i<strlen(x);i++)
    {
        strcpy(aux+i,aux+i+1);
        cout<<aux<<endl;
        strcpy(aux,x);
    }
}

```

21. Se citește de la tastatură un număr natural. Să se afișeze cel mai mare număr care se poate forma cu cifrele distincte ale numărului dat.

Ex. Pentru numărul 29363, mulțimea cifrelor distincte este {2,3,6,9}, iar numărul cerut este 9632

```

#include <iostream.h>
int ap[10],c;
void main()
{
    long n;
    cin>>n;
    int i;
    while(n) // cat timp p diferit de 0
    {
        c= n%10;
        ap[c]++;
        n=n/10;
    }
    for(i=9;i>=0;i--)
        if(ap[i]!=0)
            cout<<i;
    cout<<endl;
}

```

22. Scrieți un program care citește de la tastatură două numere naturale  $n$  și  $m$  și scrie în fișierul text „DATE.TXT” toate numerele prime din intervalul deschis  $(n,m)$ . Numerele se scriu în ordine crescătoare, câte 10 numere pe fiecare linie a fișierului, numerele fiind despărțite între ele prin câte un spațiu.

Ex. Pentru  $n=87$  și  $m=241$ , fișierul „DATE.TXT” va conține

```

89 97 101 103 107 109 113 127 131 137
139 149 151 163 167 173 179 181 191
193 197 199 211 223 227 229 233 239

```

```

#include <fstream.h>
#include "functii.txt"
// #include <math.h>
void main()

```

```

{
    int n,m,cate=0;
    cout<<"n="; cin>>n;
    cout<<"m="; cin>>m;
    fstream f("date.txt",ios::out);
    if(n%2!=0)
        n+=2;
    else
        n=n+1;
    for(int x=n;x<=m;x+=2)
        if(prim(x))
        {
            f<<x<<" ";
            cate++;
            if(cate%10==0)
                f<<endl;
        }
    f.close();
}

```

23. Pentru un numar natural n dat, afisati descompunerea lui in factori primi. Se va folosi cel putin un subprogram in rezolvarea cerintei.

Ex. Pentru n=48 se va afisa factorul 2 la puterea 4  
Factorul 3 la puterea 1.

```

#include <iostream.h>
void desc(int x)
{
    int d=2,p;
    while(x) // cat timp x diferit de 0
    {
        p=0;
        while(x%d==0)
        {
            x=x/d;
            p++;
        }
        if(p>0)
            cout<<"Factorul "<<d<<" apare de "<<p<<" ori "<<endl;
        d++;
    }
}
void main()
{
    int n;
    cout<<"n="; cin>>n;
    desc(n);
}

```

24. Pentru un numar natural n dat sa se construiasca recursiv triunghiul de numere ca in exemplul de mai jos fara a folosi nici o structura repetitiva.

1

```

1 2
1 2 3
1 2 3 4
.....
1 2 3 4 ..... n.
#include <iostream.h>
void f(int n)
{
    if(n>0)
    {
        f(n-1);
        for(int i=1;i<=n;i++)
        {
            cout<<i<<" ";
        }
        cout<<endl;
    }
}
void main()
{
    int n;
    cin>>n;
    f(n);
}

```

25. Un tablou unidimensional v contine n numere reale ordonate crescator. Sa se afiseze in ce pozitie din v se gaseste un numar real x dat. Daca nu se gaseste in v atunci sa se afiseze un mesaj corespunzator. Se va folosi o metoda eficienta de cautare.

```

#include <iostream.h>
#include "functii.txt"
int cautare(int v[],int x,int st,int dr)
{
    if(st>=dr)
        return (x==v[st])*st;
    else
    {
        int m=(st+dr)/2;
        if(v[m]==x)
            return m;
        if(x>v[m])
            return cautare(v,x,m+1,dr);
        else
            return cautare(v,x,st,m-1);
    }
}
void main()
{
    int v[50],n,x;
    citeste(v,n);
    cout<<"x="; cin>>x;
}

```

```
    if(cautare(v,x,1,n)==0)
        cout<<"nu se afla in sir";
    else
        cout<<cautare(v,x,1,n);
}
```